Introduction to Computer Organization – Fall 2023

Homework 1

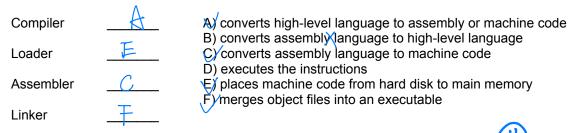
Due: @11:55pm, Monday September 25th on Gradescope

Name:	Yuzhen Cl	hen	Uniqname: _	yuzhench
	•			7

- 1. Submit a pdf of your typed or handwritten homework on Gradescope.
- 2. Your answers should be neat, clearly marked, and concise. Typed work is recommended, but not required unless otherwise stated. Show all your work, and state any special or non-obvious assumptions you make.
- 3. You may discuss your solution methods with other students, but the solutions you submit must be your own.
- 4. Late Homework Policy: Submissions turned in by 1:00 am the next day will be accepted but with a 5% penalty. Assignments turned in between 1:00 am and 11:55 pm will get a 30% penalty, and any submissions made after this time will not be accepted.
- 5. When submitting your answers to Gradescope you need to indicate what page(s) each problem is on to receive credit. The grader may choose not to grade the homework if answer locations are not indicated.
- 6. The last two questions are group questions.
 - You will turn those questions in separately and may do it in a group of up to two students (yes, you can do it by yourself if you wish).
 - <u>It is an honor code violation if a student is listed as contributing who did not actually participate in working on that problem.</u>
 - Further, we suggest that you not split this up but rather work on the problem as a group.

1. Multiple choice and the like. [15 points]

a. In the blank provided write the letter which best defines the associated term [4]



b. If each activation record for "combination" (stack frame for the function) requires 256 bytes of storage, and the maximum stack size possible on a machine is 3000 bytes, identify *all* of the following recursive function calls that will not be able to execute on the machine. [6]

c. Executing which of the following lines of LC2K assembly clears register 1 (that is sets register 1 equal to 0?) assuming register 0 is a zero? [5]

n hor o

```
A. nor 1 1 1 1 X
B. nor 0 0 1 X
C. lw 0 1 0 X
D. .fill 1 X
E. add 1 0 0 X

— C <----- Answer(s) go here
```

Problem 2 (15 points): Numbers and stuff

- 1. Convert the following numbers to 8-bit binary two's complement representation. [4]
 - a. 34 (decimal) -6432 1b 8 4 2 1

06 00100010

b. -100 (decimal)

- 2. Convert the following 16-bit binary 2's complement numbers to decimal. [4]
 - a. $\frac{-8421}{1100}$ $\frac{8+21}{1110}$ $\frac{9\times16}{1001}$ $9\times16^{\circ}+2\times16^{\prime}+14\times16^{2}+\cdots+4\times16^{3}=9+32+3584-16384=-12759$

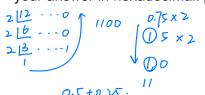
$$6 \times 16^{\circ} + 1 \times 16^{\prime} + 13 \times 16^{2} = 6 + 16 + 3328 = 3350$$

- 3. Convert the following 16-bit binary numbers to hex. [2]
 - ABCDEF a. 1101 1000 1110 1011

DX D8 EB

b. 0010 1010 0011 1010

4. Provide the 32-bit representation of -12.75 as an IEEE floating point number. Provide your answer in hexadecimal. [5]



DXC14C0000

Problem 3 (15 Points): Moving data between memory and registers

For the following LEGv8 assembly code, indicate the final state of the registers and memory locations listed once this code has finished. Assume any memory address outside of that displayed in the initial memory state contains 0.

	Initial Register State	Initial Memory State		
Register	Value	Address	Value	
X0	0x100/	0x00001000	0x4	
х1	0x1000	0x00001001	0x12	
X2	OXE DXIZD4	0x00001002	0×33 D×04	
х3	0x41FF	0x00001003	-0x2E DX 17	
Х4	0x1002 0x2E33/204	0x00001004	OXAL DXFF	
X5	0x0	0x00001005	0x77	

Fill in the following table. Express your final answers in hexadecimal. Assume memory is organized in a **little endian** manner.

Final Register State		Final Memory State		
Register	Value	Address	Value	
X0	0×1001	0x00001000	UX4	
X1	0 X 1000	0x00001001	OXIZ	
Х2	0×1204	0x00001002	0×04	
х3	0×41FF	0x00001003	DXIZ	
Х4	DX2 E 331204	0x00001004	ox F.F.	
х5	OXD	0x00001005	DX77	

Problem 4 (15 points): Arrays

For problem 4 follow the C/C++ order of operations. All variables and arrays are initialized. You may not modify other registers.

Variable to register mappings

Starting addresses

int a - r1

int b - r2

Go - r3

Blue - r4

Convert the following C code to 4 lines of LC2K assembly. Use r6 to hold the value of Blue[8], r7 to hold the value of the sum of b and Blue[8], and r8 r6 (again) to hold the any results needed to compute Go[a+4] $V_b = V_4 + 8$

Go[a+4] = b + Blue[8];

lw 4 6 8 // BlueT8] storein b.

add $2 6 7 11 r_1 = r_2 + r_6 = b + Blue[8]$ add $1 3 6 11 r_6 = r_3 + r_1 \rightarrow r_6 = G_0[a]$

SW 6 7 4 // store b+ Blue[8] to Go [at4]

Problem 5 (15 points): Floating point--mind the gap

With integers, the "gap" between representations is always 1. This means that rounding a decimal to an integer can introduce an error of up to ∓ 0.5 .

- a) In a 32-bit IEEE floating point number, what is the smallest number greater than 2 that can be exactly represented? What is the gap between that number and 2? Briefly justify your answers. [4]
- ① Smallest number > 2 that can be exactly represented; $2:1.0\times2^{1}$ 0 10000000, 100.0000, if we want to let it become a little bit bigger then 0, 10000000000, 0000000000. The gap is $0^{-23}\times2=1\times2^{-2}$
 - then \bigcirc 1000000, 100 \bigcirc 001. The gap is $2^{-23} \times 2 = 1 \times 2^{-2}$ regative exponent mantissa
- b) Assuming we are always rounding to the nearest value we can exactly represent (rounding down in the case of ties), what is the largest value we can add to 2 and get a result that is still 2? [3]
- max value is $\frac{1}{2}x2^{-24}x2^{-1} + x2^{-24}$ Since we can assume the 33th bit could be 2^{24} and also we have I in exponent and we always rounding to the nearest value.

c) Redo part a) but for 12.75 rather than 2. [3]

d) Briefly explain how the gap between floating point numbers might cause a programmer difficulties. In particular, consider a for loop where small values are being added to a larger number many thousands of times. [3]

If we write a for loop and odd a really small value to a very large number a. Then, may the value of a will not charge for several loop. Since the gas is bigger than the update value and rounding the value down.

Problem 6 (15 points, group): Transistor Performance

You must **type** the answer to these questions. Read about Moore's Law and Dennard Scaling on Wikipedia (or some other source if you wish) then answer the following questions. Each answer should be under 100 words.

- 1. Define Moore's Law in your own words.
- 2. Define Dennard Scaling in your own words.
- 3. Imagine you have a microprocessor that follows Dennard Scaling perfectly. If you double the number of transistors on the chip while keeping the area constant, *roughly* what would you expect would happen to the performance and power consumption of the microprocessor? (We are looking for things like "go up", "go down" "stay the same" etc.) Briefly explain your reasoning.
- 4. Say that the power consumed by a single transistor is constant and you double the number of transistors on the chip while keeping the area the same. *Roughly* what would you expect would happen to the performance and power consumption of the microprocessor? (We are looking for things like "go up", "go down" "stay the same" etc.) Briefly explain your reasoning.

Problem 7 (10 points, group): Writing LC2K code

Consider the following C structure.

```
struct LinkedList {
    int element;
    struct LinkedList *next;
};
```

The following LC2K2 code is supposed to sum the value of all of the elements in the linked list. The pointer of the last element of the linked list points to location zero, and no other pointer points to that location. Fill in the missing gaps with the correct values to cause this program to work. [10 points]

```
reg3 = (reg0 + stant) =
                                                         pointe : r3
      lw
                   start Load pointer to 1st element.
               0 4 reg 4 = 0+0=0 | Sum = 0 at beginning
      add 0
addLL beq 3 0
                 done
               3
      beq 0
               0
                   addLL
done halt
        .fill first,
start
                       First element of the link list
first
        .fill 4
        .fill second pointer to the next element
second .fill 10
                      (More of the data structure follows)
.....
```